CPSC 1155 – Assignment 6

C++ Arrays

Learning Objectives

The goal of this assignment is to program Arrays in C++.

Readings

You should be reading Chapters 7 and 8 of the textbook. The lectures and labs will provide additional supporting material.

Instructions

For each **Problem Statement**, follow the steps below:

1. Read the problem statement and clarify the problem.
    a. Break the problem into smaller problems if needed.
2. Determine the IPO.
    a. Determine input, output, intermediate variables, constants, conditions, and repetitions.
    b. Declare the variables and constants (data type + meaningful names).
    c. Work out the problem by hand using typical input values. Determine the range of valid input values.
    d. Determine the process.
3. Write a function as required. For each function:
    a. Determine the parameters and their data types.
    b. Determine the data type for the returned value.
4. Write the test program (main) as required. Reading input values and displaying results happen in the test program.
    a. Add comments where needed. Make sure to use a comments header to reflect the intention of your program and name of the author (you) and the date the program was written.
    b. Test, debug, and execute the program using typical values.

Submit according to the instruction in the "Submission" section.

Problem Statements

1. (password.cpp) You are required to generate a password of 10 random characters with at least 2 uppercase letters, 2 digits, and one symbol at random indices; the rest are lowercase letters.

    Write a **function** that receives two parameters as the start character and end character and generates a random character between these two characters inclusive.

    Write a **test program** that uses the above function to generate a password with the above requirements. Look for the possible symbols in the ASCII table.

2. (duplicate.cpp) Write a **pseudocode** and **C++ program** that reads in 10 numbers. As the program reads a number, it stores the number in an array only if it's not a duplicate of a number already read. The program prints the array of nonduplicate values. Here is a sample run of the program:

```
Enter 10 numbers: 21 45 33 21 4 12 6 4 33 76
The distinct numbers are: 21 45 33 4 12 6 76
```

3. (Occurrence of numbers) Write a **C++ program** that declares and initializes an integer array of size 100 with random integers between 1 and 10 using initializeArray function and counts the occurrence of each number in the array and prints the occurrences.

   Note: Print time or times correctly.

   Hint: Declare another array called count that stores the number of each value as you count the numbers of the original array. Use printArray to print the values of count.

   Here is a sample run of the program:

```
1 occurs 1 time
2 occurs 15 times
3 occurs 9 times
4 occurs 12 times
5 occurs 11 times
6 occurs 8 times
7 occurs 13 times
8 occurs 7 times
9 occurs 9 times
10 occurs 15 times
```

4. (histogram.cpp) Write a **pseudocode** and **C++ program** that reads 10 numbers from an array (use another function to initialize the array with random values between 1 and 20 inclusive) and graphs the information in the form of a bar chart or histogram—each number is printed, then a bar consisting of that many asterisks is printed beside the number. Here is a sample run of the program:

```
Element   Value      Histogram
0         12         ************
1         5          *****
2         3          ***
3         16         ****************
4         7          *******
5         6          ******
6         9          ********
7         1          *
8         15         ***************
9         10         **********
```

5. (largest.cpp) Write the following **void function** that finds the location of the largest element in a two-dimensional array.

```
void locateLargest(const double a[][4], int location[]);
```

   The location will be stored in a one-dimensional array called location that contains two elements. These two elements indicate the row and column indices of the largest element in the two-dimensional array. Write a **test**

program that prompts the user to enter a 3 * 4 two-dimensional array and displays the location of the largest element in the array. **Result must be displayed in the main function**. Here is a sample run:

```
Enter the array:
23.5   35    2    10
 4.5    3   45   3.5
  35   44  5.5   9.6
The location of the largest element is at (1, 2)
```

6. (consecutive.cpp) Write the following **function** that tests whether the array has four consecutive numbers with the same value.

```
bool isConsecutiveFour (int values [][4])
```

Write a **test program** that prompts the user to enter a series of positive integers and displays if the series contains four consecutive numbers with the same value. Your program should first prompt the user to enter the input size (i.e., the number of values in the series). Assume the maximum number of values is 40. Here are sample runs:

```
Enter the number of values: 8
Enter the values: 3 4 5 5 5 5 4 5
The list has consecutive fours
```

```
Enter the number of values: 9
Enter the values: 3 4 5 8 5 5 4 4 5
The list has no consecutive fours
```

7. (nine.cpp) Nine coins are placed in a 3 x 3 matrix with some face up and some face down. You can represent the state of the coins using a 3 x 3 matrix with values 0 (head) and 1 (tail). Here are some examples:

```
000     101     110     101     100
010     001     100     110     111
000     100     001     100     110
```

Each state can be represented using a binary number. For example, the preceding matrices correspond to the following numbers:

000010000 101001100 110100001 101110100 100111110

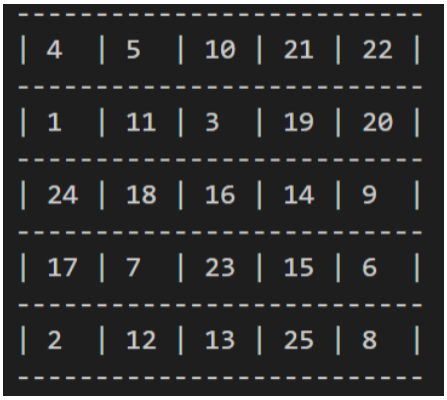
The total number of possibilities is 512. So, you can use decimal numbers 0, 1, 2, 3, …, and 511 to represent all states of the matrix. Write a **C++ program** that prompts the user to enter a number between 0 and 511 and displays the corresponding binary number and matrix with characters H and T. Make sure binary number is stored in an integer array. Here is a sample run:

```
Enter a number between 0 and 511: 299
The binary number is: 100101011
The matrix is:
THH
THT
HTT
```

8. (board.cpp) Write the following function to receive an empty 5 x 5 two-dimensional array and put numbers from 1 to 25 in the array. Each number could only **occur once**. The location of each number should be **random**. Use a **seed value of 233 in srand() function**.

```
void createBoard(int board [][5])
```

Write another display function to receive a two-dimensional array and display it like the way below.



Write a **test program** that uses function *createBoard* to create an array and use the display function to print it.

Hint: Basically, there are two ideas to generate the board. One is firstly generating an array from 1 to 25 in order, and then swapping each number with another number at a randomly generated location. Another is when a number is generated, checking it with the already generated numbers. If there is any duplicate, generate a new number again until there is no duplicate. The first idea is recommended.

Submission

Submit a zip folder named as yourname_assign6.zip to Brightspace. This folder should consist the **C++ codes** in individual .cpp files and one pseudocode.txt file with all your **pseudocode** (for questions 2 and 4).

Please make sure that all your .cpp files compile and run properly before submission. Your file must run properly in order to receive full marks.

Marking Scheme

There are 10 marks for each question with the following details:

Question 1: password.cpp

- 6 for correct function
- 4 for correctly generated password

Question 2: duplicate.cpp

- 6 for correct solution
- 4 for pseudocode

Question 3: occurrence.cpp

- 10 for correct solution

Question 4: histogram.cpp

- 4 for correct solution
- 2 for format
- 4 for pseudocode

Question 5: largest.cpp

- 10 for correct solution

Question 6: consecutive.cpp

- 10 for correct solution

Question 7: nine.cpp

- 6 for correct binary array
- 4 for correct display result

Question 8: board.cpp

- 6 for correct createBoard function
- 4 for correct display function